MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1603 January, 1997
C.B.C.L. Paper No. 145

# Image-based View Synthesis

**Shai Avidan, Theodoros Evgeniou, Amnon Shashua, Tomaso Poggio**

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu. The pathname for this publication is: ai-publications/1500-1999/AIM-1603.ps.Z

**Abstract**

We present a new method for rendering novel images of flexible 3D objects from a small number of example images in correspondence. The strength of the method is the ability to synthesize images whose viewing position is significantly far away from the viewing cone of the example images ("view extrapolation"), yet without ever modeling the 3D structure of the scene. The method relies on synthesizing a chain of "trilinear tensors" that governs the warping function from the example images to the novel image, together with a multi-dimensional interpolation function that synthesizes the non-rigid motions of the viewed object from the virtual camera position. We show that two closely spaced example images alone are sufficient in practice to synthesize a significant viewing cone, thus demonstrating the ability of representing an object by a relatively small number of model images — for the purpose of cheap and fast viewers that can run on standard hardware.



This report describes research done within the Artificial Intelligence Laboratory and the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences at the Massachusetts Institute of Technology. This research is sponsored by grants from ONR under contracts N00014-96-1-0342 and N00014-93-1-0385; by a grant from ONR/ARPA under contract N00014-92-J-1879 by the National Science Foundation under contract ASC-9217041 (this award includes funds from ARPA provided under the HPCC program) and by grants ACTS AC 074 and VANGUARDUS-IS94-00120.

# 1 Introduction

In this paper we develop a reprojection technique for synthesizing novel views of a 3D object, given a collection of 2D model images in correspondence. We also consider the synthesis of novel views from a single model image and develop a method to control non-rigid transformations of the viewed object, such as facial expressions in the case of faces.

The most significant aspect of our approach is the ability to synthesize images that are far away from the viewing positions of the sample model images without ever computing explicitly any 3D information about the scene. This property provides a multi-image representation of the 3D object using a minimal number of images. In our experiments, for example, two closely spaced frontal images of a face are sufficient for generating photo-realistic images from viewpoints within a 60 degrees cone of visual angle – further extrapolation is possible but the image quality degrades. The immediate application of our results is to provide a very fast 3D viewing system based on a small number of images that can run on standard hardware.

The notion of image-based rendering is gaining momentum in both the computer graphics and computer vision communities. The general idea is to achieve photo-realistic virtual images while avoiding the computational-intensive process of acquiring a 3D model followed by rendering. Instead, one seeks to use a number of model images of the 3D object or scene as a representation from which novel views can be synthesized *directly* by means of image warping.

The forerunner of this approach is to create a panoramic image of a scene (mosaic) from overlapping images taken from a fixed location while varying the orientation of the camera. The mosaic is mapped to a virtual cylinder that allows the user to look continuously at all directions but not to move. This is the basis for the QuickTimeVR system [7].

The fixed position constraint can be relaxed by computing the optical flow between the example images and using it to interpolate between the cylinders constructed at different locations (cf. [4, 5, 6]) (originally proposed for views, not mosaics, but the principle is the same). However, interpolation may produce physically-invalid images. Seitz and Dyer [22] proposed a physically-valid view interpolation method. The method involves recovering the epipolar geometry between the two acquired images and having interpolation done along the rectified epipolar lines.

Interpolation can also be performed directly on the plenoptic function [1] which represents the amount of light emitted at each point in space as a function of direction. Levoy et al. [16] and Gortler et al. [11] interpolate between a dense set of several thousands of example images to reconstruct a reduced plenoptic function (under an occlusion-free world assumption). They considerably increase the number of example images to avoid computing optical flow between the model images.

The major limitations of the aforementioned techniques is that a relatively large number of model images is required to represent an object. The alternative approach, along the lines of this paper, is to reduce the number of acquired (model) images by exploiting the 3D-from-2D geometry of the problem with the aid of corresponding points between the model images. Laveau and Faugeras [15] were the first to use the epipolar constraint for view synthesis, allowing them to extrapolate, as well as interpolate, between the example images. Epipolar constraints, however, are subject to singularities that arise under certain camera motions (like when the virtual camera center is collinear with the centers of the model cameras) and the relation between translational and rotational parameters of the virtual camera and the epipolar constraint is somewhat indirect and hence requires the specification of matching points [15]. The singular camera motions can be relaxed by using the depth map of the environment. McMillan and Bishop [19] use a full depth map (3D reconstruction of the camera motion and the environment) together with the epipolar constraint to provide a direct connection between the virtual camera motion and the reprojection engine. Depth maps are easily provided for synthetic environments, whereas for real scenes the process is fragile especially under small base-line situations that arise due to the requirement of dense correspondence between the model images/mosaics [12].

In this paper we propose a new view-synthesis method that makes use of the recent development of multi-linear matching constraints, known as trilinearities, that were first introduced in [23]. The trilinearities provide a general (not subject to singular camera configurations) warping function from model images to novel synthesized images governed directly by the camera parameters of the virtual camera. Therefore, we provide a true multi-image system for view synthesis that does not require a companion depth map, nor the full reconstruction of camera parameters among the model cameras, yet is general and robust. The strength of our method is demonstrated by the ability to work with closely spaced acquired images yet synthesize high-quality views at a significant extrapolation from the viewing angles of the acquired images. Furthermore, our method can be generalized to work with a single acquired model image and to allow non-rigid transformations by integrating multi-linear constraints and multi-dimensional interpolation.

The main contributions of our work are:

1. The introduction of the trilinear tensor as the warping function.

2. The derivation of a tensorial operator which is the heart of the method. The operator generates a cascading set of tensors from two model views in correspondence and the parameters of the virtual camera motion. The tensorial operator does not require the estimation of the baseline between the acquired images (typically a fragile process)

thereby enabling the model images to be closely spaced without hindering much the robustness of the synthesis process.

3. The combination of the tensor with a learning method for the generation of virtual views of an object given only a single model image.

4. The combination of rigid transformations using the tensor with non-rigid transformations achieved with multi-dimensional view interpolation.

On the experimental side, we have tested the proposed method on a variety of objects with a variety of cameras in real-world conditions where neither camera calibration (or even camera type) is available nor the lighting conditions are controlled. We demonstrated that correspondence is practical for closely spaced images, and that the synthesis method is sufficiently accurate and robust.

# 2 View Synthesis in Tensor Space

The view synthesis approach is based on the following paradigm. Three views satisfy certain matching constraints of a trilinear form, represented by a tensor. Thus, given two views in correspondence and a tensor, the corresponding third view can be generated uniquely by means of a warping function, as described below in more detail. We describe how to recover the tensor parameters and show a "driver" function that governs the change in tensor coefficients as a result of moving the virtual camera.

## 2.1 The Trilinear Warping Function

The trilinear tensor concatenates together the camera transformation matrices (camera locations) across three views, as follows. Let $P$ be a point in 3D projective space projecting onto $p, p', p''$ in three views $\psi, \psi', \psi''$ respectively, represented by the two dimensional projective space. The relationship between the 3D and the 2D spaces is represented by the $3 \times 4$ matrices, $[I, 0]$, $[A, v']$ and $[B, v'']$, i.e.,
$p = [I, 0]P$,p' = [A,v']P$and$p" = [B,v"]P
where $A, B$ stand for the rotational component of camera motion (generally these are 2D homography matrices) and $v', v''$ stand for the translational component (generally these are the epipolar points).

We may adopt the convention that $p = (x, y, 1)^\top$, $p' = (x', y', 1)^\top$, $p'' = (x'', y'', 1)^\top$ and, thus, $P = (x, y, 1, \rho)$. The coordinates $(x, y), (x'y'), (x'', y'')$ are matching points across the three images.

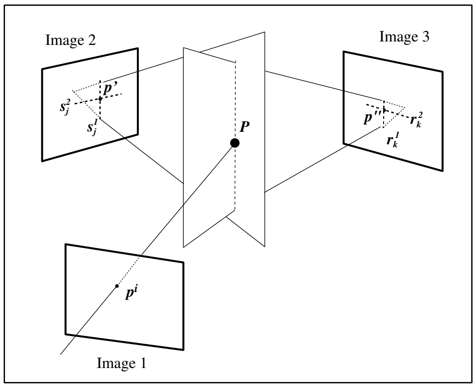



Figure 1: Each of the four trilinear equations describes a matching between a point $p$ in the first view, some line $s_j^\mu$ passing through the matching point $p'$ in the second view and some line line $r_k^\rho$ passing through the matching point $p''$ in the third view. In space, this constraint is an intersection between a ray and two planes.

The trilinear tensor is an array of 27 entries:

$$\alpha_i^{jk} = v'^j b_i^k - v''^k a_i^j. \qquad i,j,k = 1,2,3 \tag{1}$$

where the covariant-contravariant indexing notation is assumed (see Appendix A). The tensor $\alpha_i^{jk}$ forms the set of coefficients of certain trilinear forms that vanish on any corresponding triplet $p, p', p''$:

$$p^i s_j^\mu r_k^\rho \alpha_i^{jk} = 0 \tag{2}$$

where $s_j^\mu$ are any two lines ($s_j^1$ and $s_j^2$) intersecting at $p'$, and $r_k^\rho$ are any two lines intersecting at $p''$ (see Fig. 1).

Since each of the free indices $\mu, \rho$ is in the range 1,2, we have 4 trilinear equations which are unique up to linear combinations. If we choose the canonical form where $s$ and $r$ represent vertical and horizontal lines, then the four trilinear forms,referred to as trilinearities, are expanded as follows:

$$\begin{aligned}
x''\boldsymbol{\alpha}_i^{13}p^i - x''x'\boldsymbol{\alpha}_i^{33}p^i + x'\boldsymbol{\alpha}_i^{31}p^i - \boldsymbol{\alpha}_i^{11}p^i &= 0,\\
y''\boldsymbol{\alpha}_i^{13}p^i - y''x'\boldsymbol{\alpha}_i^{33}p^i + x'\boldsymbol{\alpha}_i^{32}p^i - \boldsymbol{\alpha}_i^{12}p^i &= 0,\\
x''\boldsymbol{\alpha}_i^{23}p^i - x''y'\boldsymbol{\alpha}_i^{33}p^i + y'\boldsymbol{\alpha}_i^{31}p^i - \boldsymbol{\alpha}_i^{21}p^i &= 0,\\
y''\boldsymbol{\alpha}_i^{23}p^i - y''y'\boldsymbol{\alpha}_i^{33}p^i + y'\boldsymbol{\alpha}_i^{32}p^i - \boldsymbol{\alpha}_i^{22}p^i &= 0.
\end{aligned}$$

Since every corresponding triplet $p, p', p''$ contributes four linearly independent equations, then seven corresponding points across the three views uniquely determine (up to

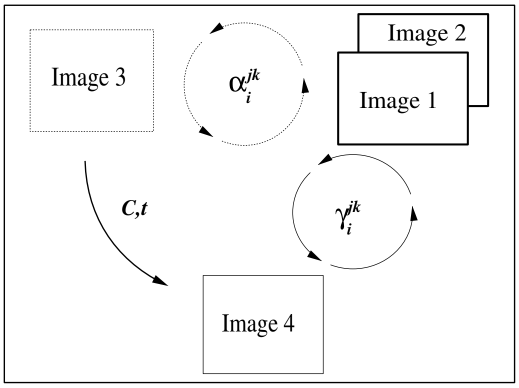



Figure 2: We generate tensor $\gamma_i^{jk}$, that relates images $1, 2$ with some novel image 4, from the previous tensor $\alpha_i^{jk}$ (dashed) and the virtual camera motion parameters $(C, t)$ from image 3 to image 4. Tensor $\alpha_i^{jk}$ relates images $1, 2$ and 3 and is computed only once at the pre-processing stage. Tensor $\gamma_i^{jk}$ is computed every time the user specifies a new $(C, t)$. We use tensor $\gamma_i^{jk}$ to render the novel image (image 4) from example images $1, 2$.

scale) the tensor $\alpha_i^{jk}$. These constraints first became prominent in [23] and the underlying theory has been studied intensively in [27, 13, 24, 10, 28, 14, 25].

One can readily see that given two views in full correspondence and the tensor (recovered using 7 matching points with a third view), the entire third view can be synthesized by means of forward warping. From each trilinearity we can simply extract either $x''$ or $y''$, thus for every matching pair $p, p'$ we can obtain $p''$. We then copy at $p''$ the appropriate brightness value, for example the average of the pixel values at $p$ and $p'$ in the two model images. This process is referred to as "reprojection" in the literature. There are alternative ways of performing reprojection, but if we would like to do it without recovering first a 3D model of the scene, the trilinear tensor generally provides the best results since it is free from singular configurations (see [2, 23, 26]).

We have described so far the implementation of the reprojection paradigm via the trilinear equations. In other words, given two model views and a tensor, the third view is uniquely determined and can be synthesized by means of a warping function applied to the two model images. In image-based rendering we would like to obtain the tensor via user specification of the location of a virtual camera, rather than by the specification of (at least) seven matching points. This is described next.

## 2.2 The basic Tensorial Operator

The basic tensorial operator describes how to modify (transform) a tensor so as to represent a new configuration of three cameras. We are particularly interested in the case where only one camera has changed its position and orientation. Thus, by repeated application of the operator on a seed tensor with a sequence of desired virtual camera positions (translation and orientation) we obtain a chain of warping functions (tensors) from the set of acquired images (from which the seed tensor was computed) to create the desired virtual views (see Fig. 2).

Consider the tensor $\alpha_i^{jk}$ of the views $< 1,2,3 >$ (in that order), and assume the user wishes to apply an incremental change of position of the third image, i.e., rotate the third camera position by the $3 \times 3$ coordinate matrix $C$, and translate it by the $3 \times 1$ translation vector $t$ — this motion would result to a novel view, call it view 4. Then the tensor $\gamma_i^{jk}$ of the views $< 1,2,4 >$ is given by:

$$\gamma_i^{jk} = v'^j(c_l^k b_i^l) - (v''^l c_l^k + t^k)a_i^j = c_l^k \alpha_i^{jl} - t^k a_i^j. \tag{3}$$

This is so because we use Eq. 1 where we replace the motion parameters $v''^k$,$b_i^k$ from the first image to the third image, with $(v''^l c_l^k + t^k), (c_l^k b_i^l)$ which depend on the motion parameters from the first image to the novel one. The matrix $a_i^j$ representing the rotational component of camera motion between the two model views 1,2 can be represented in closed form as a function of the tensor $\alpha_i^{jk}$ as described in [21]:

$$\begin{aligned}
\Omega_X &= det\begin{pmatrix} \alpha_2^{j3} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} /K \\
\Omega_Y &= det\begin{pmatrix} -\alpha_1^{j3} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} /K \\
\Omega_Z &= det\begin{pmatrix} \alpha_1^{j2} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} /K \\
K &= det\begin{pmatrix} \alpha_2^{j2} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix}
\end{aligned} \tag{4}$$

where $\Omega_X, \Omega_Y, \Omega_Z$ are rotation angles and $\alpha_2^{j2}$ stands for $(\alpha_2^{12},\ \alpha_2^{22},\ \alpha_2^{32})$, etc.

To summarize, Eq. 3 is a general formula for transforming the tensor based on an incremental camera motion of a fixed (third) camera. Therefore, starting from a "seed"

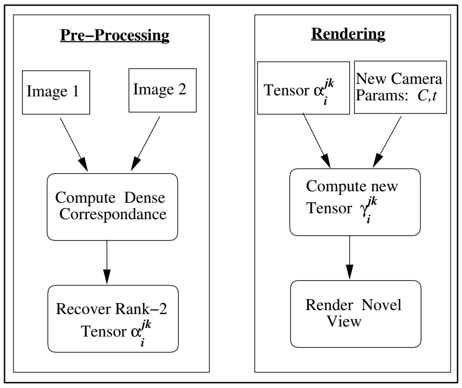



Figure 3: View synthesis is divided into two parts. The pre-processing stage, done only once and the actual rendering done for every image.

tensor and a sequence of desired camera motions, the set of corresponding tensors can be generated and used to warp the acquired images onto the novel views. We next consider how to obtain the seed tensor that starts the process.

## 2.3 The Seed Tensor of Two Views

Given two acquired images we can construct a special tensor composed of the elements of the fundamental matrix [9] that can serve as a seed tensor that starts the chain of tensors, as follows. Consider a configuration of three views in which views 2,3 coincide, i.e., Eq. 1 becomes:

$$\gamma_i^{jk} = v'^j a_i^k - v'^k a_i^j \qquad (5)$$

where $\gamma_i^{jk}$ is the tensor of the image triplet $< 1, 2, 2 >$. It can be readily verified that the elements of $\gamma_i^{jk}$ are composed of the fundamental matrix $f_{ij} = \epsilon_{ikl} v'^k a_j^l$, $-f_{ij}$, and the remaining (nine) elements vanish. It will not be shown here, but the rank of $\gamma_i^{jk}$ is 2 whereas the rank of the tensor of three distinct views is 4 — but otherwise all other properties remain and, in particular, $\gamma_i^{jk}$ can serve as the first tensor that starts the synthesis process described above.

## 2.4 The View Synthesis Loop

We are ready to describe the full synthesis loop starting from a pair of closely spaced acquired (model) images of a 3D object. The method is divided into two stages — a preprocessing stage, done only once, and the actual rendering done for every new frame (see Fig. 3).

1. **Preprocessing**

   (a) Compute dense correspondence (optical flow) between the two model images. We use a coarse-to-fine implementation of [18] described in [3]. Interactive tools for improving correspondence [17] can be used as well, although in our experiments optical flow alone was sufficient.

   (b) Recover the fundamental matrix of the two model images from the correspondences. We use a robust estimator based on a Monte-Carlo technique described in [20].

   (c) Construct the rank-2 tensor (Eq. 5) from the elements of the fundamental matrix.

2. **View Synthesis**

   (a) Accept camera motion parameters (rotation and translation) from the second camera to its new position.

   (b) Apply Eq. 3 to compute the tensor of the two model images and the novel one.

   (c) Synthesize the novel view by (forward) warping the two model views using the tensor computed in the previous step.

### 2.5 Experiments

We conducted four experiments, two of them shown here in Figs. 4, 5 and 6 and the remaining two are shown in the companion Video (and web page). In the "statue" sequence (Fig. 4) a pair of images of an African statue was captured with an SGI Indy Cam. The image size was $260 \times 480$ pixels.

In the "Shannon" sequence (Figs. 5 and 6) the images were taken at a studio but no camera calibration was performed. The image size was $620 \times 764$ pixels. Due to the quality of the images we were able to extrapolate to over 90 degrees from the viewing cone defined by the example images. This is shown in Fig. 6. As expected, the quality of the optical flow determines the quality of the rendered image, especially when going to large extrapolations. Occlusions also become a significant source of noise especially along the boundaries. However, in all cases degradation in image quality is graceful.

## 3 Synthesis from a single model view

In the previous section, we have discussed how to synthesize new images for different viewpoints, given two examples images. Suppose now that only one image of a specific object, say $\mathbf{p}_{nov}$, is available. Does our reprojection method break down? This section sketches a solution to this question. The idea is to create an additional – virtual – example image of object $\mathbf{p}_{nov}$ from just one real view of it. Once one obtains a second image of the object from a different viewpoint, one can use the reprojection algorithm of section 2.4 to generate subsequent virtual images. To accomplish this task without using a parametric 3D model, one may start from a collection of example views of another similar object $\mathbf{p}$ which plays the role of a prototype for representing generic transformations of the object class that the two objects belong to. Faces form such a class of objects. In general, we want to generate from one 2D view $\mathbf{Img}_{nov}$ of a 3D object $\mathbf{p}_{nov}$ other views, exploiting knowledge of views of other objects of the same class. This idea of generating "virtual" views of an object by using class-specific knowledge has been discussed before (see references in [5]). Suppose that we have two views $\mathbf{Img}_{ref}$ and $\mathbf{Img}_{p}$ of the prototype. We take $\mathbf{Img}_{ref}$ to appear in the same pose as $\mathbf{Img}_{nov}$. $\mathbf{Img}_{p}$ is a slightly transformed (i.e., rotated) view of $\mathbf{Img}_{ref}$ (see diagram in Fig. 7). We can then compute the optical flow $\mathbf{S}_p$ between these two views. Moreover, since the prototype object $\mathbf{p}$ is assumed to be "similar" to object $\mathbf{p}_{nov}$, we assume that we can find good correspondence $\mathbf{S}_{nov}$ between $\mathbf{Img}_{ref}$ and $\mathbf{Img}_{nov}$. We subsequently generate the optical flow $\mathbf{S}_{p+nov}$ between the view $\mathbf{Img}_{ref}$ and a new view of the object $\mathbf{p}_{nov}$ by the vector addition:

$$\mathbf{S}_{p+nov} = \mathbf{S}_p + \mathbf{S}_{nov}. \tag{6}$$

A new view, $\mathbf{Img}_{p+nov}$, of object $\mathbf{p}_{nov}$ is then rendered by texture mapping from the single available view $\mathbf{Img}_{nov}$ after forward warping from $\mathbf{Img}_{ref}$ using optical flow $\mathbf{S}_{p+nov}$. In a sense, we "map" the learned transformation (optical flow $\mathbf{S}_p$) from $\mathbf{Img}_{ref}$ to $\mathbf{Img}_{nov}$ using flow $\mathbf{S}_{nov}$. We now have two images of object $\mathbf{p}_{nov}$ that our reprojection technique can use to simulate a virtual camera and generate new images and image sequences.

We demonstrate this technique in Fig. 7 using as an example a self-portrait of Van Gogh. A slight rotation is learned from another similar prototypical "object", in this case another face, to generate a first virtual image of Van Gogh. Then the reprojection method of section 2.4 is used to generate subsequent views.

# 4 Incorporating Non-rigid Transformations

So far we have described a technique that allows the user to generate new images by controlling the rigid degrees of freedom that correspond to motion of the camera. From two or more images in correspondence it may also be possible to generate new images of non-rigid 3D objects as a function of input parameters that correspond to non-rigid transformations such as a change in expression of a face. The underlying operation is multidimensional interpolation, a simple extension of traditional image morphing. We outline the technique and illustrate how it can be integrated with the algebraic method described so far.

Let us assume that $n$ images are available and that pixel-wise correspondence can be computed with an optical flow algorithm between one of them, chosen as the "zero" reference image, and each of the $n-1$ others. As we saw earlier, correspondence associates to each image $i$ the optical flow, that we note as $\mathbf{S}_i$, of the position of each pixel relative to the reference image. We can also associate to each image a vector of color values, the "texture" vector, that we note as $\mathbf{T}_i$. The texture vector $\mathbf{T}_i$ is simply the image $i$ warped to the shape of the reference image by the optical flow $\mathbf{S}_i$. Let us also assume that the user defines the values $\mathbf{r}_i$ of the non-rigid parameters of interest to be associated with each one of these "example" images.

A multidimensional interpolation technique such as Radial Basis Functions or splines is then used to interpolate the $n$ example pairs $(\mathbf{r}_i, (\mathbf{S}_i, \mathbf{T}_i))$ (see for instance [5]). The mapping from the input space of non-rigid parameters to the output space of images, expressed in terms of textures and flows, is provided by the following interpolation scheme which can be regarded as a learning network ([4, 5])

$$\begin{aligned} \mathbf{S}(\mathbf{r}) &= \textstyle\sum_{i=1}^{n} \mathbf{c}_i G(\mathbf{r}-\mathbf{r}_i), \\ \mathbf{T}(\mathbf{r}) &= \textstyle\sum_{i=1}^{n} \mathbf{a}_i G(\mathbf{r}-\mathbf{r}_i), \end{aligned} \tag{7}$$

where the $\mathbf{c}_i$ and $\mathbf{a}_i$ are vectors of coefficients, and $G$ is a basis function, which may be a radial basis function, like the Gaussian or a spline, like a tensor product spline. The network coefficients $\mathbf{c}_i$ and $\mathbf{a}_i$ are found by solving the linear system of equations (7) over the training data ([4, 5]).

Given a new vector $\mathbf{r}$ of non-rigid parameters, the network of Eq. 7 synthesizes a new $(\mathbf{S}, \mathbf{T})$ using the learned coefficients $\mathbf{c}_i$ and $\mathbf{a}_i$, which is then rendered in a new image by warping $\mathbf{T}_i$ according to the warping field $\mathbf{S}_i$, effectively performing multidimensional morphing.

This simple technique can be used to control several non-rigid degrees of freedom such as facial expressions, as shown by [4, 5, 17, 8]. It can be combined directly with the algebraic technique described earlier to control the position of the virtual camera. Thus, given a set of images with different non-rigid parameters from different viewpoints (for each viewpoint we have an image for each of the non-rigid parameters), the algorithm is composed of the following steps:

**Preprocessing:**

1. For each of the viewpoints (rigid parameters) choose a reference image and compute dense correspondence (optical flow) between this reference and the other images representing the non-rigid parameters. Obtain the texture vectors for each of these images – relative to the corresponding reference image.

2. For each of the viewpoints use the example images to learn the mapping from the non-rigid parameters to the flow and texture vectors found in the previous step.

3. For one of the non-rigid parameters (any one will do) follow the preprocessing steps of the algorithm described in section 2.4.

**View Synthesis:**

1. Accept the values of the rigid and non-rigid parameters.

2. For the desired set of values of the non-rigid parameters use the multidimensional interpolation algorithm to generate a virtual image for each of the viewpoints.

3. Apply the reprojection algorithm described in section 2.4 using the accepted values of the rigid parameters and the images generated in step 2 to obtain an image from the desired viewpoint.

Fig. 8 shows the simple case of one non-rigid degree of freedom. Given four images, corresponding to two values of the non-rigid parameter and two viewpoints, a virtual image for the desired intermediate expression is obtained by interpolation for each of the viewpoints. Then our reprojection technique generates views of the intermediate expression from the desired new viewpoint.

# 5 Conclusions

The method we describe in this paper can render novel images of flexible 3D objects from a small number of example images without the need of an explicit 3D model. Its main strength is the ability to synthesize images whose viewing position is significantly far away from the viewing cone of the example images.

Clearly the key step in this class of techniques is the computation of pixel-wise correspondence between the example images. We addressed this problem by using an optical flow algorithm from the computer vision literature that estimates dense sets of pixel-level correspondences. It is well known that correspondence is a very difficult problem which can be solved only for images that are similar enough and do not suffer from significant self-occlusions. 3D model-based approaches, however, suffer from even worse correspondence problems, if the 3D models are themselves estimated from a set of images. Our technique has a major advantage relative to others, at least for the rigid degrees of freedom of the camera, since it relies on pairs of images with a *small* baseline, which helps the critical correspondence stage. Methods that estimate 3D structure are very noisy with small baselines. Morphing techniques, such as [22], require large baselines since they cannot perform extrapolation.

The problems with correspondence and occlusions can be solved at the expense of increasing the number of examples. An apparently deeper problem is how does the approach to synthesis scale with the number of pose/expression parameters? It would seem that increasing the number of parameters used to control image synthesis may require an exponential increase in the number of example images. Fortunately, there are several reasons why the problem is not as bad as it may seem. First, our technique needs only two images from close viewpoints to deal with all the rigid degrees of freedom. Second, the number of examples required for the non-rigid degrees of freedom may remain quite low, for appropriate choices of the input parameters (see [5] and references therein). Third, it is possible to bypass the curse of dimensionality problem by representing an object as a hierarchy of components. Interpolation networks responsible for each component are thus independent, for instance the mouth separately from the eyes [17, 8]. The main factors likely to set a lower bound on the number of example images needed for a given set of control parameters are therefore simple visibility constraints: every part of the scene to be rendered must be visible in at least two of the example images.

## References


[1] Adelson, E.H. and J.R. Bergen The Plenoptic Function and the Elements of Early Vision In *Computational Models of Visual Processing*, Chapter 1, Edited by Michael Landy and J. Anthony Movshon. The MIT Press, Cambridgem, Mass. 1991.

[2] E.B. Barrett, P.M. Payton, and G. Gheen. Robust algebraic invariant methods with applications in geometry and imaging. In *Proceedings of the SPIE on Remote Sensing*, San Diego, CA, July 1995.

[3] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, Santa Margherita Ligure, Italy, June 1992.

[4] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. A.I. Memo No. 1431, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, October 1993.

[5] D. Beymer, and T. Poggio. Image Representations for Visual Learning. *Science*, 272, pages 1905–1909, 1996.

[6] S.E. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH*, pages 279–288, Anahiem, CA, August 1993.

[7] Shenchang Eric Chen. QuickTimeVR - an image-based approach to virtual environment navigation. In *SIGGRAPH*, pages 29–38, 1995.

[8] T. Ezzat Example-based analysis and synthesis for images of human faces. M.Eng. Thesis, Massachusetts Institute of Technology, February 1996.

[9] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proceedings of the European Conference on Computer Vision*, pages 563–578, Santa Margherita Ligure, Italy, June 1992.

[10] O.D. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *Proceedings of the International Conference on Computer Vision*, Cambridge, MA, June 1995.

[11] Gortler, S.J., Grzeszczuk, R., Szeliski, R. and Cohen, M. The Lumigraph In *SIGGRAPH*, pages 43 - 54, 1996.

[12] W.E.L. Grimson. Why stereo vision is not always about 3D reconstruction. A.I. Memo No. 1435, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, July 1993.

[13] R. Hartley. A linear method for reconstruction from lines and points. In *Proceedings of the International Conference on Computer Vision*, pages 882–887, Cambridge, MA, June 1995.

[14] A. Heyden. Reconstruction from image sequences by means of relative depths. In *Proceedings of the International Conference on Computer Vision*, pages 1058–1063, Cambridge, MA, June 1995.

[15] S. Laveau and O.D. Faugeras. 3-d scene representation as a collection of images. In *Proceedings of the International Conference on Pattern Recognition*, pages 689–691, Jerusalem, Israel, October 1994.

[16] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, August 1996.

[17] S. Lines The photo-realistic synthesis of novel views from example images M.S. Thesis, Massachusetts Institute of Technology, May 1996.

[18] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings IJCAI*, pages 674–679, Vancouver, Canada, 1981.

[19] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH*, pages 39–46, 1995.

[20] Torr P.H.S., Zisserman A., and Murray D. Motion clustering using the trilinear constraint over three views. In *Workshop on Geometrical Modeling and Invariants for Computer Vision*. Xidian University Press., 1995.

[21] B. Rousso, S. Avidan, A. Shashua, and S. Peleg. Robust recovery of camera rotation from three frames. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San-Francisco, CA, June 1996.

[22] Steven M. Seitz, Charles R. Dyers. View morphing. In *SIGGRAPH*, pages 21–30, 1996.

[23] A. Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, 1995.

[24] A. Shashua and P. Anandan. The generalized trilinear constraints and the uncertainty tensor. In *Proceedings of the ARPA Image Understanding Workshop*, Palm Springs, CA, February 1996.

[25] A. Shashua and S. Avidan. The rank4 constraint in multiple view geometry. In *Proceedings of the European Conference on Computer Vision*, Cambridge, UK, April 1996.

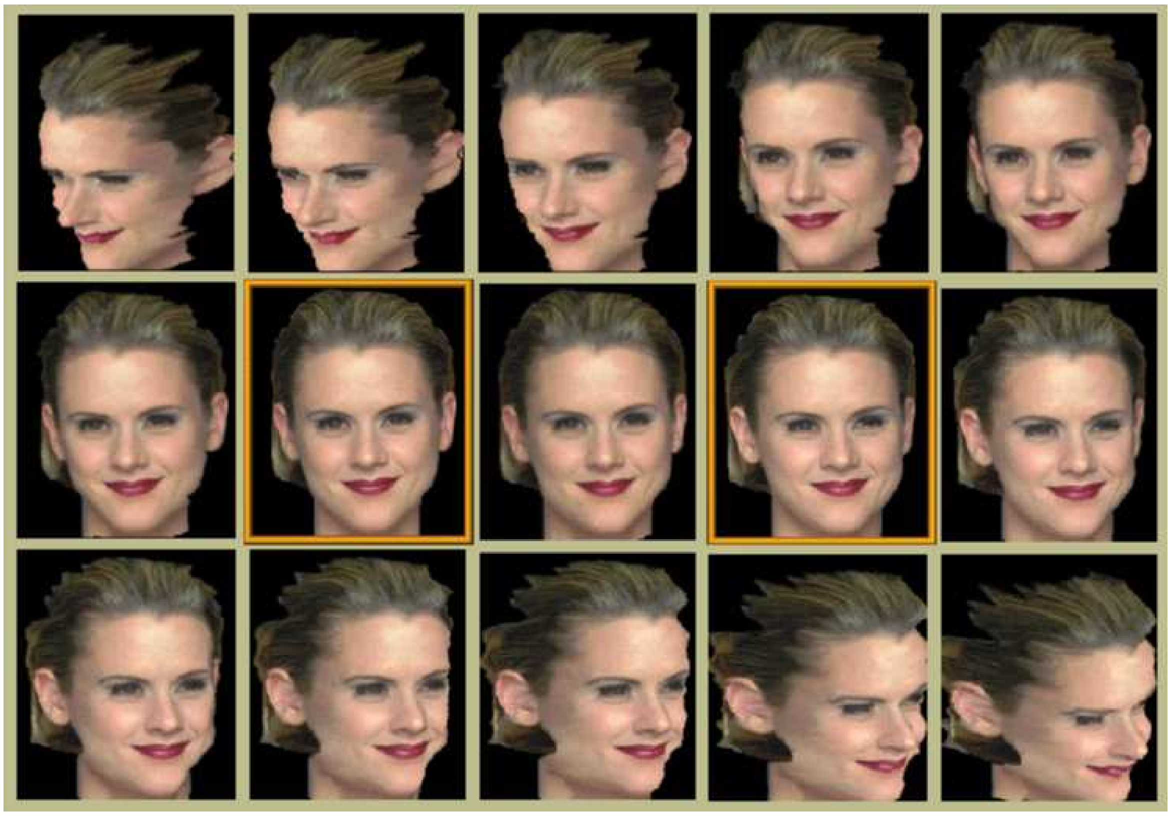
[26] A. Shashua and S.J. Maybank. Degenerate $n$ point configurations of three views: Do critical surfaces exist? Technical Report TR 96-19, Hebrew University of Jerusalem, November 1996.

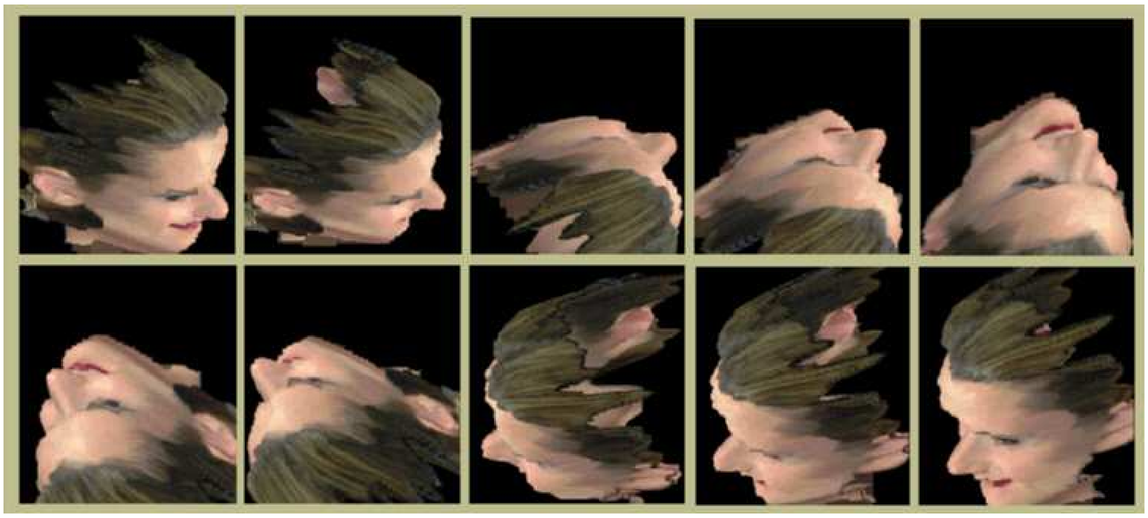
[27] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. In *Proceedings of the International Conference on Computer Vision*, June 1995.

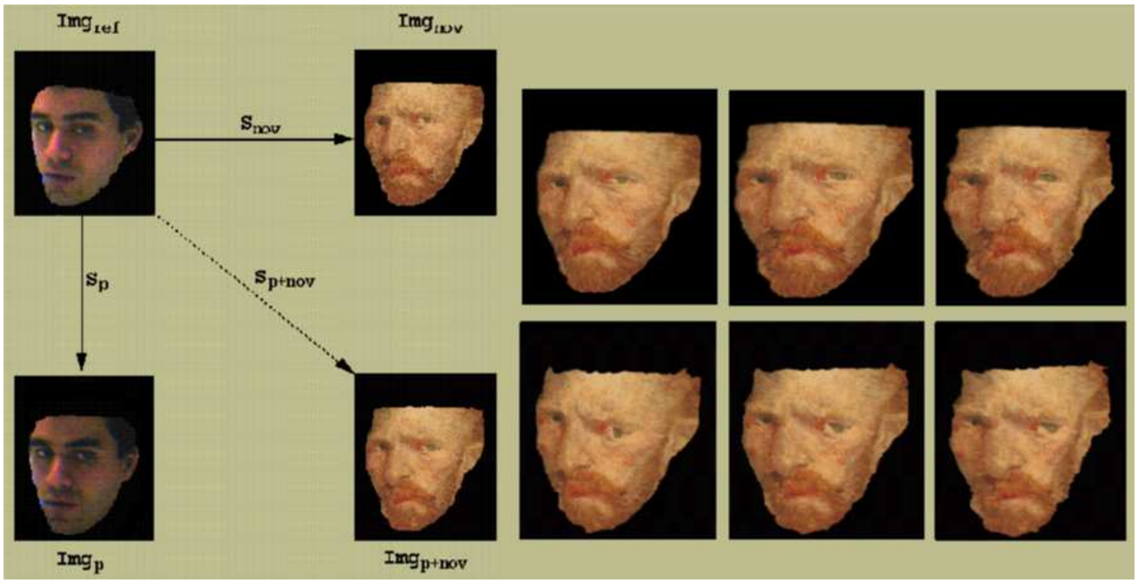
[28] B. Triggs. Matching constraints and the joint image. In *Proceedings of the International Conference on Computer Vision*, pages 338–343, Cambridge, MA, June 1995.

## A On Tensorial Notations

We use the covariant-contravariant summation convention: a point is an object whose coordinates are specified with superscripts, i.e., $p^i = (p^1, p^2, ...)$. These are called contravariant vectors. An element in the dual space (representing hyper-planes — lines in $\mathcal{P}^2$), is called a covariant vector and is represented by subscripts, i.e., $s_j = (s_1, s_2, ....)$. Indices repeated in covariant and contravariant forms are summed over, i.e., $p^i s_i = p^1 s_1 + p^2 s_2 + ... + p^n s_n$. This is known as a contraction. For example, if $p$ is a point incident to a line $s$ in $\mathcal{P}^2$, then $p^i s_i = 0$. Vectors are also called 1-valence tensors. 2-valence tensors (matrices) have two indices and the transformation they represent depends on the covariant-contravariant positioning of the indices. For example, $a_i^j$ is a mapping from points to points, and hyper-planes to hyper-planes, because $a_i^j p^i = q^j$ and $a_i^j s_j = r_i$ (in matrix form: $Ap = q$ and $A^\top s = r$); $a_{ij}$ maps points to hyper-planes; and $a^{ij}$ maps hyper-planes to points. When viewed as a matrix the row and column positions are determined accordingly: in $a_i^j$ and $a_{ji}$ the index $i$ runs over the columns and $j$ runs over the rows, thus $b_j^k a_i^j = c_i^k$ is $BA = C$ in matrix form. An outer-product of two 1-valence tensors (vectors), $a_i b^j$, is a 2-valence tensor $c_i^j$ whose $i, j$ entries are $a_i b^j$ — note that in matrix form $C = ba^\top$. The tensor of vector products is denoted by $\epsilon_{ijk}$ (indices range 1-3) operates on two contravariant vectors of the 2D projective plane and produces a covariant vector in the dual space (a line): $\epsilon_{ijk} p^i q^j = s_k$, which in vector form is $s = p \times q$, i.e., $s$ is the vector product of the points $p$ and $q$.







Img_ref
Img_nov
S_nov
S_p
S_p+nov
Img_p
Img_p+nov